**Write a Difference between HTTP1.1 vs HTTP2**

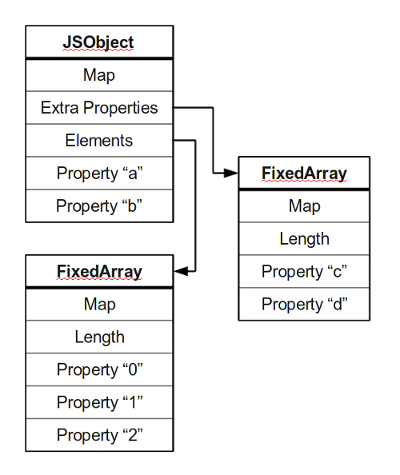
|  |  |  |
| --- | --- | --- |
| keywords | HTTP 1.1 | HTTP 2 |
| Year | 1997 | 2015 |
| Key Features | It supports connection reuse i.e. for every TCP connection there could be multiple requests and responses, and pipelining where the client can request several resources from the server at once. However, pipelining was hard to implement due to issues such as head-of-line blocking and was not a feasible solution. | Uses multiplexing, where over a single TCP connection resources to be delivered are interleaved and arrive at the client almost at the same time. It is done using streams which can be prioritized, can have dependencies and individual flow control. It also provides a feature called server push that allows the server to send data that the client will need but has not yet requested. |
| Status Code | Introduces a warning header field to carry additional information about the status of a message. Can define 24 status codes, error reporting is quicker and more efficient. | Underlying semantics of HTTP such as headers, status codes remains the same. |
| Authentication Mechanism | It is relatively secure since it uses digest authentication, NTLM authentication. | Security concerns from previous versions will continue to be seen in HTTP/2. However, it is better equipped to deal with them due to new TLS features like connection error of type Inadequate Security |
| Caching | Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags. | HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream. |
| Web Traffic | HTTP/1.1 provides faster delivery of web pages and reduces web traffic as compared to HTTP/1.0. | HTTP/2 utilizes multiplexing and server push to effectively reduce the page load time by a greater margin along with being less sensitive to network delays. |

2**. Write a blog about objects and its internal representation in Javascript**

1. Objects are complex and each object may contain a combination of primitive data-types as well as reference data-types.  
   An object is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.
2. Other way around, objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.  
   An object can be created with figure brackets {} with an optional list of properties. A property is a “key: value” pair, where a key is the property name value can be anything.

**Object & properties**

A JavaScript object has properties associated with it. A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. You access the properties of an object with a simple dot-notation: example given below.



Most objects contain all their properties in a single block of memory (“a”, and “b”). All blocks of memory have a pointer to a map, which describes their structure.

Named properties that don’t fit in an object are usually stored in an overflow array (“c”, and “d”).Numbered properties are stored separately, usually in a contiguous array.

The JavaScript standard allows developers to define objects in a very flexible way, and it is hard to come up with an efficient representation that works for everything. An object is essentially a collection of properties: basically key-value pairs. You can access properties using two different kinds of expressions:

obj.prop

obj[“prop”]

According to the spec, property names are always strings. If you use a name which is not a string, it is implicitly converted to a string. This may be a little surprising: if you use a number as a property name, it gets converted to a string as well (at least according to the spec). Because of this, you can store values at negative or fractional array indices. So a JavaScript object is basically a map from strings to values.